Doctoral Course
in
Speech Recognition

Part 1

Mats Blomberg

September-December 2003

Introduction

- Course objective
  - deeper insight into basic and specific methods and algorithms
  - *understanding* - not exact details of equations
  - no derivation of theorems and algorithms
  - Not covered
    - Phonetics, linguistics
  - Signal processing relevant parts (short time spectral analysis)
  - theory of probabilistics and pattern recognition overviewed
  - merit 5p
- Recommended background
  - GSLT course in "Speech and speaker recognition" or equivalent

Background

- Acoustic phonetics
- Speech analysis
  - Short Time Spectral Analysis
  - MFCC
- Recognition
  - Dynamic programming and DTW
  - Fundamentals of hidden Markov models
  - Viterbi decoding
  - Phoneme-based speech recognition methods

Literature

- Spoken Language Processing
  - A Guide to Theory, Algorithm and System Development
  - X. Huang, A. Acero and H-W Hon
  - Contains theoretically heavy parts and many equations but it is not necessary to follow all derivations. The verbose explanations of their functions are easier to follow.
- Separate papers
  - Finite State Transducers
  - Bayesian Networks
  - Articulatory inspired approaches
  - ...

Course organization

- 3 - 4 course one-day meetings (10.15 - 16.00)
  - #1 (19 Sep) : Introduction, Lecture 1st 1/3 of the course
  - #2 (end Oct): Lecture 2nd 1/3, discussion, HTK tutorial, exercise presentation, presentation of subjects for term paper
  - #3 (end Nov): Lecture 3rd 1/3, discussion
  - #4 (January): Students' presentation of individual term papers
- Exercises
- Term paper + review + presentation
- How many meetings and when?

Course overview

- Day #1
  - Probability, Statistics and Information Theory (pp 73-131: 59 pages)
  - Pattern Recognition (pp 133-197: 65 pages)
  - Speech Signal Representations (pp 275-336 62 pages)
  - Hidden Markov Models (pp 377-413: 37 pages)
- Day #2
  - Acoustic Modeling (pp 415-475: 61 pages)
  - Environmental Robustness (pp 477-544: 68 pages)
  - Language Modeling (pp 545-590: 46 pages)
  - Basic Search Algorithms (pp 591-643: 53 pages)
  - HTK tutorial
- Day #3
  - Large-Vocabulary Search Algorithms (pp 645-685: 41 pages)
  - Applications and User Interfaces (pp 919-956: 38 pages)
  - Other topics
- Day #4
  - Presentations of term papers

Term paper

- Choose subject from a list or suggest one yourself
- Review each others reports
- Suggested topics
  - Language models for speech recognition
  - Limitations in standard HMM and ways to reduce them
  - Pronunciation variation and their importance for speech recognition
  - New search methods
  - Techniques for robust recognition of speech
  - Own work and experiments after discussion with the teacher

Book organization 1(2)

  - Ch 1 Introduction
- Part I: Fundamental theory
  - Ch 2 Spoken Language Structure
  - Ch 3 Probability, Statistics and Information Theory
  - Ch 4 Pattern Recognition
- Part II: Speech Processing
  - Ch 5 Digital Signal Processing
  - Ch 6 Speech Signal Representation
  - Ch 7 Speech Coding

Book organization 2(2)

- Part III: Speech Recognition
  - Ch 8 Hidden Markov Models
  - Ch 9 Acoustic Modeling
  - Ch 10 Environmental Robustness
  - Ch 11 Language Modeling
  - Ch 12 Basic Search Algorithms
  - 13 Large-Vocabulary Search Algorithms
- Part IV: Text-to-Speech Systems
  - Ch 14 Text and Phonetic Analyses
  - Ch 15 Prosody
  - Ch 16 Speech Synthesis
- Part V: Spoken Language systems
  - Ch 17 Spoken Language Understanding
  - Ch 18 Applications and User Interfaces

Ch 3. Probability, Statistics and Information Theory

- Conditional Probability and Bayes' Rule
- Covariance and Correlation
- Gaussian Distributions
- Bayesian Estimation and MAP Estimation
- Entropy
- Conditional Entropy
- Mutual Information and Channel Coding

Conditional Probability and Bayes’ Rule

- Bayes’ rule - the common basis for all pattern recognition

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

$$P(A_i|B) = \frac{P(A_iB)}{P(B)} = \frac{P(B|A_i)P(A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)}$$

Example: P(A) = 0.1, P(B) = 0.08, P(B|A)=0.24
P(A|B) = 0.24*0.1/0.08 = 0.3

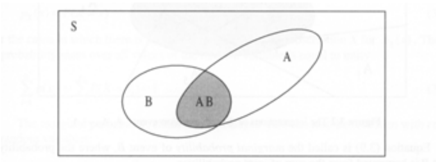


Figure 3.1 The intersection AB represents where the joint event A and B occurs concurrently.

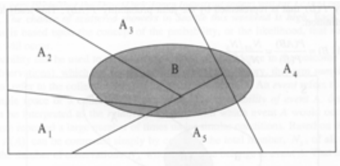


Figure 3.2 The intersections of B with partition events $A_1, A_2, \cdots, A_n$.

Sannolikhetsbaserad igenkänning

Bayes’ regel för betingade sannolikheter

$$P(Ord\ /\ Akustik) = \frac{P(Akustik\ /\ Ord) \times P(Ord)}{P(Akustik)}$$

P(Ord/Akustik) är *a posteriori sannolikheten* för en ordföljd givet den akustiska informationen.

P(Akustik/Ord) är *sannolikheten* att ordföljden genererar den akustiska informationen och beräknas i ett träningsmaterial.

P(Ord) ges av språkmodellen och är *a priori sannolikheten* för ordföljden (N-gram).

P(Akustik) kan ses som en *konstant* eftersom den är oberoende av ordföljden och kan ignoreras

Kombinerar akustisk och språklig kunskap!

Mean, Covariance and Correlation

- Mean $E(X) = \sum_x x f(x)$
- Covariance $Cov(X,Y) = E\left[(X-\mu_x)(Y-\mu_y)\right]$
- Correlation $\rho_{XY} = \dfrac{Cov(X,Y)}{\sigma_X \sigma_Y}$
- Multidimensional (Mean vector, covariance matrix)

$$E(\mathbf{X}) = \begin{bmatrix} E(X_1) \\ \vdots \\ E(X_n) \end{bmatrix} \qquad \mathbf{\Sigma_X} = Cov(\mathbf{X}) = \begin{bmatrix} Cov(X_1,X_1) & \cdots & Cov(X_1,X_n) \\ \vdots & & \vdots \\ Cov(X_n,X_1) & \cdots & Cov(X_n,X_n) \end{bmatrix}$$

Gaussian Distributions

- One-dimensional

$$f(x\,|\,\mu,\sigma^2) = N(\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

- Multivariate n-dimensional

$$f(\mathbf{X}=\mathbf{x}\,|\,\boldsymbol{\mu},\mathbf{\Sigma}) = N(x;\boldsymbol{\mu},\mathbf{\Sigma}) = \frac{1}{(2\pi)^{n/2}\,|\,\mathbf{\Sigma}\,|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

3.2 Estimation theory

- The basis for training a speech recogniser
- Estimate parameters of a probability distribution function
  - Minimum/Least Mean Squared Error Estimation
    - Minimize the difference between the distribution of the data and the model
  - Maximum Likelihood Estimation
    - Find the distribution with the maximum likelihood of generating the data
  - Bayesian Estimation and MAP Estimation
    - Assumes that we have a prior distribution that is modified by the new data

Minimum/Least Mean Squared Error Estimation

- Modify a model of the distribution to approximate the data with minimum error
- Find a function that predicts the value of Y from having observed X
- Estimation is made on joint observations of X and Y
- Minimize: $E(Y-\hat{Y})^2 = E(Y-g(X))^2$
- Minimum Mean Squared Error (MMSE) when the joint distribution is known
- Least Squared Error (LSE) when the distribution is unknown, only observation pairs (Ex. curve fitting)
- MMSE and LSE becomes equivalent with infinite number of samples

Maximum Likelihood Estimation (MLE)

- The most widely used parametric estimation method
- Find the distribution that maximizes the likelihood of generating the observed data

$$\mathbf{\Phi}_{MLE} = \arg\max_{\Phi} p(\mathbf{x} \mid \mathbf{\Phi})$$

- Corresponds to intuition
  - Max likelihood is achieved when the model has the same distribution as the observed data
- Example: univariate Gaussian pdf

$$\mu_{MLE} = \frac{1}{n}\sum_{k=1}^{n} x_n = E(x) \qquad \sigma^2_{MLE} = \frac{1}{n}\sum_{k=1}^{n}(x_k - \mu_{MLE})^2 = E\left[(x_k - \mu_{MLE})^2\right]$$

Bayesian Estimation and MAP Estimation

- Assumes that we have a prior distribution that is modified by the new data
- Use Bayes' rule to find the new posterior distribution Φ

$$\mathbf{\Phi}_{MAP} = \arg\max_{\Phi} p(\mathbf{x} \mid \mathbf{\Phi})\, p(\mathbf{\Phi})$$

- Univariate Gaussian Mean: $\rho = \frac{\sigma^2 \mu + n\nu^2 \bar{x}_n}{\sigma^2 + n\nu^2}$ Var: $\tau^2 = \frac{\sigma^2 \nu^2}{\sigma^2 + n\nu^2}$
- MAP: Maximum A Posteriori probability is a Bayesian Estimator
- MAP becomes MLE with uniform prior distribution or infinite number of training data
- Valuable for limited training data and for adaptation

Entropy and Perplexity

- Information in seeing event x_i with probability $P(x_i)$: $I(x_i) = {}^2\log \frac{1}{P(x_i)}$
- Entropy is the average information over all possible x values

$$H(X) = E[I(X)] = \sum_S P(x_i) I(x_i) = \sum_S P(x_i)^2 \log \frac{1}{P(x_i)} = -\sum_S P(x_i)^2 \log(P(x_i))$$

- Perplexity $PP(X) = 2^{H(X)}$
  - The equivalent size of an imaginary list with equi-probable words
  - Perplexity for English letters: 2.39, English words: 130
- Conditional Entropy
  - Input X is distorted by a noisy channel into output Y
  - Example: Confusion matrix

$$H(X|Y) = -\sum_X \sum_Y P(X = x_i, Y = y_j)^2 \log P(X = x_i | Y = y_j)$$

3.4.4 Mutual Information and Channel Coding

- Mutual Information I(X;Y): The difference between the entropy of X and the conditional entropy of X given Y
- The average difference between the number of bits required to specify X outcome when Y is not known and when Y is known
- If X and Y independent: I = 0

$$I(X;Y) = H(X) - H(X \mid Y) = \ldots = E\left[\log \frac{P(X,Y)}{P(X)P(Y)}\right]$$

Ch 4. Pattern Recognition 1(3)

- Bayes' Decision Theory
  - Minimum-Error-Rate Decision Rules
  - Discriminant Functions
- How to Construct Classifiers
  - Gaussian Classifiers
  - The Curse of Dimensionality
  - Estimating the Error Rate
  - Comparing Classifiers (McNemar's test)

Pattern Recognition 2 (3)

- Discriminative Training
  - Maximum Mutual Information Estimation
  - Minimum-Error-Rate Estimation
  - Neural networks
- Unsupervised Estimation Methods
  - Vector Quantization
  - The K-Means Algorithm
  - The EM Algorithm
  - Multivariate Gaussian Mixture Density Estimation

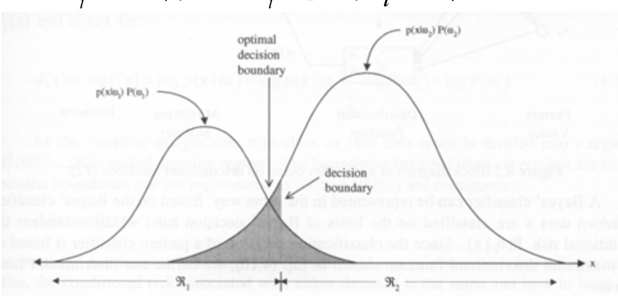
Pattern Recognition 3 (3)

- Classification and Regression Trees (CART)
  - Choice of question set
  - Splitting criteria
  - Growing the tree
  - Missing values and conflict resolution
  - Complex questions
  - The Right-Sized Tree

4.1.1 Minimum-Error-Rate Decision Rules

- Bayes' decision rule
- The decision is based on choosing the candidate that maximizes the posterior probability (results in minimum decision error)

$$k = \underset{i}{\arg\max}\, P(\omega_i \mid x) = \underset{i}{\arg\max}\, p(x \mid \omega_i) P(\omega_i)$$



4.1.2 Discriminant Functions

- The decision problem viewed as classification problem
  - Classify unknown data into one of *s* known categories
  - Using *s* discriminant functions
- Minimum-error-rate classifier:
  - Maximize a posteriori probability: Bayes' decision rule
- For two-class problem:
  - Likelihood ratio: $\ell(x) = \frac{p(x \mid \omega_1)}{p(x \mid \omega_2)}$ $\quad \ell(x) > T : \omega_1 \quad \ell(x) < T : \omega_2$ $\quad T = \frac{P(\omega_2)}{P(\omega_1)}$
- Fig 4.3 Decision boundaries

Discriminant Functions

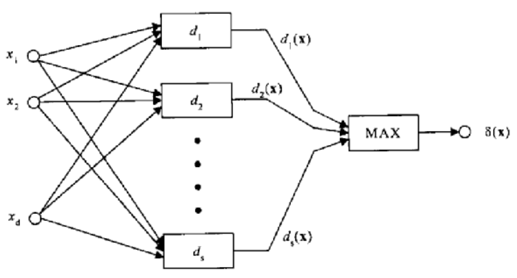
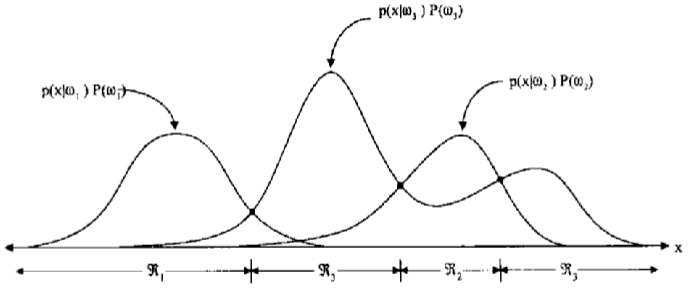


Fig 4.2 A classifier based on discriminant functions

Decision boundaries

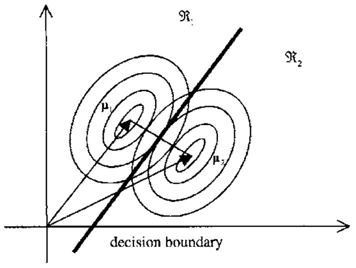


4.2.1 Gaussian classifiers

- The class-conditional probability density is assumed to have a Gaussian distribution

$$p(\mathbf{x}\,|\,\omega_i) = \frac{1}{(2\pi)^{d/2}\,|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]$$

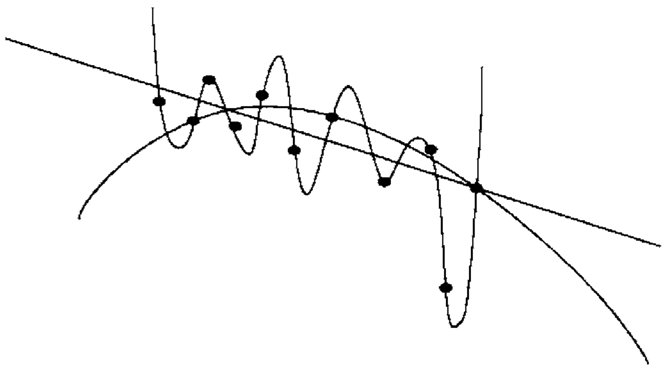
- Decision boundary



4.2.2 The Curse of Dimensionality

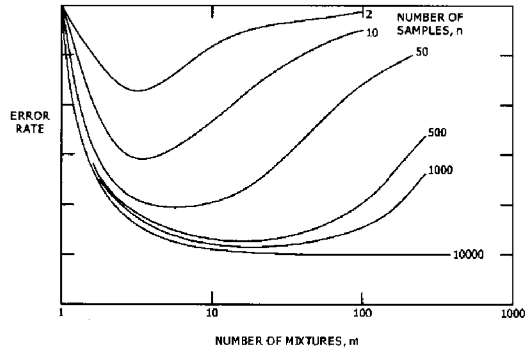
- More features (e.g. higher dimensions or more parameters in density function) lead (in theory) to lower classification error rate
- In practice: may lead to worse results due to too little training data
- Paradox called *The curse of dimensionality*
- Fig 4.6 Curve fitting
- Fig 4.7 Phoneme classification

The curse of dimensionality: curve fitting



The curse of dimensionality: Two-phoneme classification

Error rate as a function of the number of Gaussian mixture densities and the number of training samples



4.2.3 Estimating the Error Rate

- Computation from parametrical model has problems (error under-estimation, bad model assumptions, very difficult)
- Recognition error on training data is a lower bound (Warning!)
- Use independent test data
- How to partition the available speech data
  - Holdout method
  - V-fold cross validation (Leave-one-out method)

4.2.4 Comparing classifiers

- McNemar's test
  - Compares two classifiers by looking at samples where only one made an error

		Q_2 Correct	Q_2 Incorrect
Q_1	Correct	N_{00}	N_{01}
	Incorrect	N_{10}	N_{11}

$n = N_{01} + N_{10}$ N_{xx} has binomial distribution $B(n, 1/2)$

Test the null hypothesis that the classifiers have the same error rates (z-test)

4.3 Discriminative Training

- Maximum Likelihood Estimation models each class separately, independent of other classes

- Discriminative Training aims at models that maximize the discrimination between the classes
  - Maximum Mutual Information Estimation (MMIE)
  - Minimum-Error-Rate Estimation
  - Neural networks

4.3.1 Maximum Mutual Information Estimation (MMIE)

- Discriminative criterion:
  - For each model to estimate, find a setting that maximizes the probability ratio between the model and the sum of all other models
- Maximize $$\frac{p(\mathbf{x}|\omega_i)p(\omega_i)}{\sum_{k \neq i} p(\mathbf{x}|\omega_k)p(\omega_k)}$$
- Gives different result compared to MLE. MLE maximizes the numerator only
- Theoretically appealing but computationally expensive
  - Every sample used for all classes
  - Gradient descent algorithm

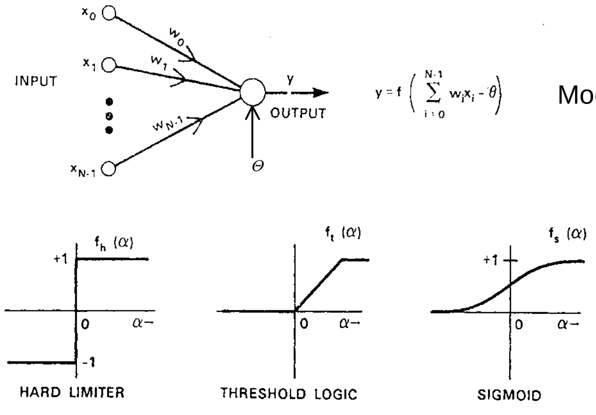
4.3.2 Minimum-Error-Rate Estimation

- Also called Minimum-classification-error (MCE) training, discriminative training,
- Iterative procedure (gradient descent)
  - Re-estimate models, classification, improve correctly recognized models and suppress mis-recognized models
- Computationally intensive, used for few classes
- Corrective training
  - Simple and faster error-correcting procedure
  - Move the parameters of the correct class towards the training data
  - Move the parameters of the near-miss class away from the training data
  - Good results

4.3.3 Neural Networks

- Inspired by nerve cells in biological nervous systems
- Many simple processing elements connected to a complex network.
- Single-Layer Perceptron Fig. 4.10
- Multi-Layer Perceptron (MLP) Fig 4.11
  - Back propagation training

Artificiella NeuronNät - ANN



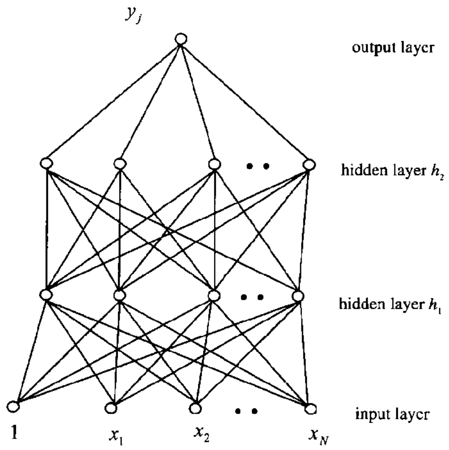
Modell av nervcell

Figure 1: Computation performed in a single node. Three representative nonlinearities are shown.

4.3.3 Multi-Layer Perceptron

$$\mathbf{y} = sigmoid(\mathbf{W'}_y\,(\mathbf{h}_2))$$

$$y_j = \frac{1}{1+e^{-(w_{0j}+\sum_i w_{ij}h_{2i})}}$$



The Back Propagation Algorithm

ALGORITHM 4.1: *THE BACK PROPAGATION ALGORITHM*

Step 1: Initialization: Set $t = 0$ and choose initial weight matrices $\mathbf{W}$ for each layer. Let's denote $w_{ij}^k(t)$ as the weighting coefficients connecting i^{th} input node in layer $k-1$ and j^{th} output node in layer k at time t.

Step 2: Forward Propagation: Compute the values in each node from input layer to output layer in a propagating fashion, for $k = 1$ to K

$$v_j^k = sigmoid(w_{0j}(t) + \sum_{i=1}^{N} w_{ij}^k(t) v_i^{k-1}) \quad \forall j \tag{4.72}$$

where $sigmoid(x) = \frac{1}{1+e^{-x}}$ and v_j^k is denoted as the j^{th} node in the k^{th} layer

Step 3: Back Propagation: Update the weights matrix for each layer from output layer to input layer according to:

$$\bar{w}_{ij}^k(t+1) = w_{ij}^k(t) - \alpha \frac{\partial E}{\partial w_{ij}^k(t)} \tag{4.73}$$

where $E = \sum_{i=1}^{s} \| y_i - o_i \|^2$ and $(y_1, y_2, \ldots y_s)$ is the computed output vector in Step 2.

α is referred to as the learning rate and has to be small enough to guarantee convergence. One popular choice is $1/(t+1)$.

Step 4: Iteration: Let $t = t+1$. Repeat Steps 2 and 3 until some convergence condition is met.

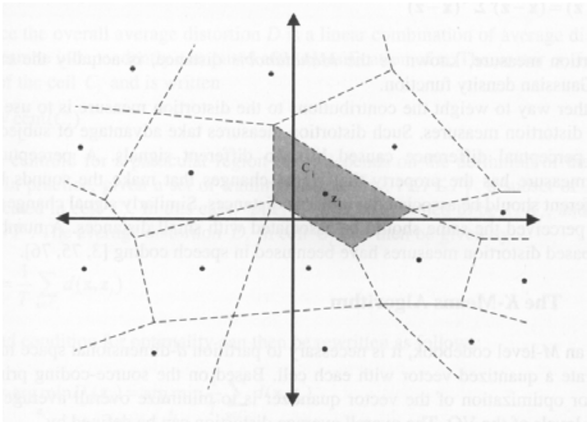
4.4 Unsupervised Estimation Methods

- Vector Quantization
- The EM Algorithm
- Multivariate Gaussian Mixture Density Estimation

4.4.1 Vector Quantization (VQ)

- Described by a codebook, a set of prototype vectors (codewords)
- An input vector is replaced by the index of the codeword with the smallest distortion
- Distortion Measures
  - Euclidean
  - Mahalanobis distance
- Codebook generation algorithms
  - The K-Means Algorithm
  - The LBG Algorithm

Vector Quantization



Partitioning of a two-dimensional space into 16 cells

The K-Means Algorithm

- 1. Choose an initial division between the codewords
- 2. Classify each training vector into one of the cells by choosing the closest codeword
- 3. Update all codewords by computing the centroids of the training vectors
- 4. Repeat steps 2 and 3 until the distortion ratio between current and previous codebooks is above a preset threshold

- Comment
  - Converges to *local* optimum
  - Initial choice is critical

The LBG Algorithm

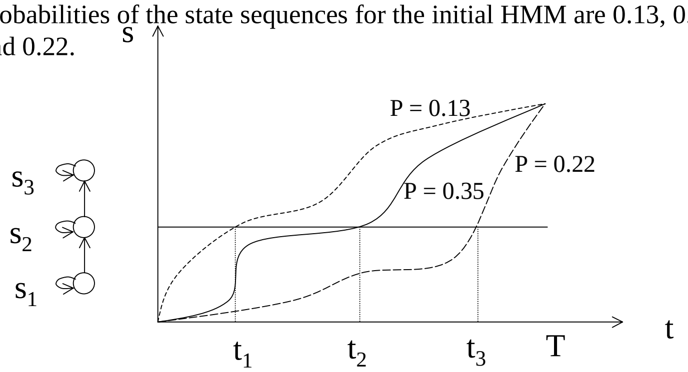
- 1. Initialization.
  - Set number of cells M = 1. Find the centroid of all training data.
- 2. Splitting.
  - Split M into 2M by finding two distant points in each cell. Set these as centroids for 2M cells.
- 3. K-Means Stage.
  - Use K-Means algorithm to modify the centroids for minimum distortion.
- 4. Termination
  - If M equals the required codebook size, STOP. Otherwise go to 2.

4.4.2 The Expectation Maximization (EM) Algorithm

- Used for training of hidden Markov models
- Generalisation of Maximum-Likelihood Estimation
- Problem approached
  - Estimate distributions (ML) of several classes when the training data is not classified (e.g. into states of the models)
  - Is it possible to train the classes anyway? (Yes - *local* maximum)
- Simplified iterative procedure (similar to K-Means procedure for VQ)
  - 1. Initialise class distributions
  - 2. Using current parameters, compute the class probability for each training sample.
  - 3. Each sample updates *each* class distribution by the probability weights
    - Maximum-likelihood estimate of distributions, replace current distr.
  - 4. Repeat 2+3 until convergence (Will converge)

Simplified illustration of EM estimation

Say, three paths have been found in a training utterance. The probabilities of the state sequences for the initial HMM are 0.13, 0.35 and 0.22.



New $E(s_2) = (0.13\ X(t_1) + 0.35\ X\ (t_2) + 0.22\ X(t_3))\ /\ 0.70$

Not as simple as it may look, though

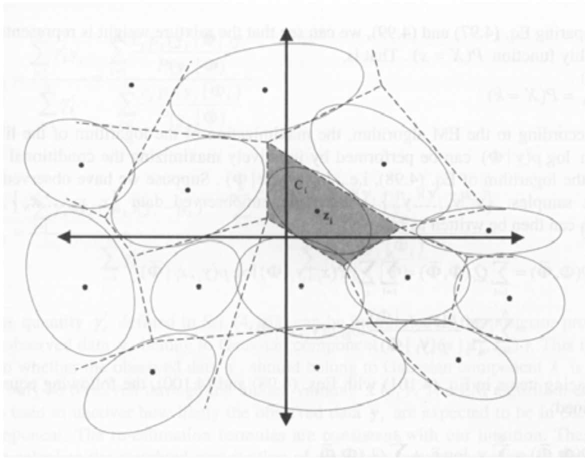
4.4.3 Multivariate Gaussian Mixture Density Estimation

- Probability density is weighted sum of Gaussians:

$$p(\mathbf{y} \mid \mathbf{\Phi}) = \sum_{k=1}^{K} c_k p_k(\mathbf{y} \mid \mathbf{\Phi}_k) = \sum_{k=1}^{K} c_k N_k(\mathbf{y} \mid \boldsymbol{\mu}_k, \mathbf{\Sigma}_k)$$

  - c_k is the probability of component k, $c_k = P(X = k)$
- Analogy
  - GM - VQ,
  - EM algorithm - K-means algorithm
  - VQ minimizes codebook distortion; GM maximizes the likelihood of the observed data
  - VQ performs hard assignment; EM performs soft assignment

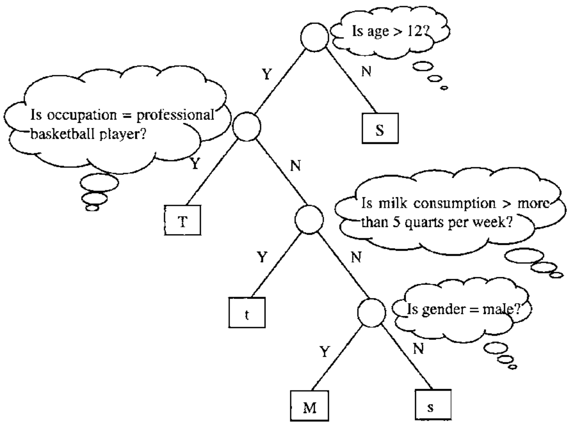
Partitioning space into Gaussian density functions



4.5 Classification and Regression Trees (CART)

- Binary decision tree
- An automatic and data-driven framework to construct a decision process based on objective criteria
- Handles data samples with mixed types, nonstandard structures
- Handles missing data, robust to outliers and mislabeled data samples
- Used in speech recognition for model tying

Binary tree structure for height classification



Steps in constructing a CART

- 1. Find set of questions
- 2. Put all training samples in root
- 3. Recursive algorithm
  - Find the best combination of question and node. Split the node into two new nodes
  - Move the corresponding data into the new nodes
  - Repeat until right-sized tree is obtained
- Greedy algorithm, only locally optimal, splitting without regard to subsequent splits
  - Dynamic programming would help but computationally heavy
  - Works well in practice

4.5.1 Choice of Question Set

- Can be manually selected
- Automatic procedure:
- Simple (singleton) - complex questions
  - Simple questions about a single variable
- Discrete variable questions
  - Does x_i belong to set S? S is any possible subset of the training samples
- Continuous variable questions
  - Is $x_i <= c_n$? c_n is midpoint between two training samples

4.5.2 Splitting Criteria

- Find the pair of node and question for which split gives
  - Discrete variable
    - Maximum reduction in entropy

$$\Delta\overline{H}_t(q) = \overline{H}_t(Y) - (\overline{H}_l(Y) + \overline{H}_r(Y))$$

  - Continuous variables
    - The maximum gain in likelihood

$$\Delta\overline{L}_t(q) = L_1(\mathbf{X}_1|N) + L_2(\mathbf{X}_2|N) - L_X(\mathbf{X}|N)$$

  - For regression purposes
    - The largest reduction in squared error from a regression of the data in the node

4.5.3 Growing the Tree

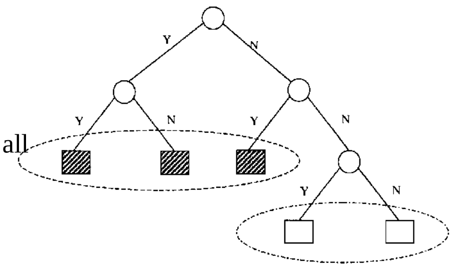
- Stop growing a node when either
  - All samples in the node belong to the same class
  - The greatest entropy reduction falls below threshold
  - The number of data samples in the node is too small

4.5.4 Missing Values and Conflict Resolution

- Missing values
  - ?
- Conflict resolution
  - Two questions may achieve the same entropy reduction and the same partitioning
  - One question may be sub-question to the other
  - Select the sub-question (since more specific)

4.5.5 Complex Questions

- Problem
  - Simple (one-variable) questions may result in similar leaves in different locations
  - Over-fragmenting
- Solution
  - Form composite questions by all possible combinations of the simple-question leaf nodes in the tree



4.5.6 The Right-Sized Tree

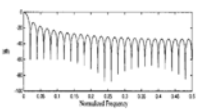
- Too many splits improves classification on training data but reduction on test data (Curse of dimensionality)
- Use a pruning strategy to gradually cut back the over-grown tree until the minimum misclassification on the test data is achieved
  - Minimum Cost-Complexity Pruning
    - Produces a sequence of trees with increased pruning
  - Select the best tree using either of
    - Independent Test Sample Estimation (fixed test data)
    - V-fold Cross Validation (train on (v-1) parts, test on 1, circulate)

Ch 5 Digital Signal Processing

- Digital Signals and Systems
- Continuous Frequency transforms
  – The Fourier Transform
  – Discrete-Frequency Transforms
    • The Discrete Fourier Transform (DFT)
    • The Fast Fourier Transform (FFT)
- Digital Filters and Windows
  – Rectangular, Hamming and Hanning window functions

Window Functions

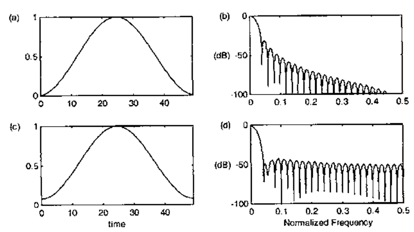
- The frequency spectrum of the window function affects the signal spectrum
- Rectangular Window
  – discontinuities at boundaries smear the spectrum



- Hamming and Hanning windows

$$h_h[n] = \begin{cases} (1-\alpha) - \alpha\cos(2\pi n/N) & 0 \le n \le N \\ 0 & otherwise \end{cases}$$

- α = 0.5 Hanning window
- α = 0.46 Hamming window



5.3.3 The Discrete and Fast Fourier Transforms (DFT & FFT)

- The Discrete Fourier Transform (DFT)

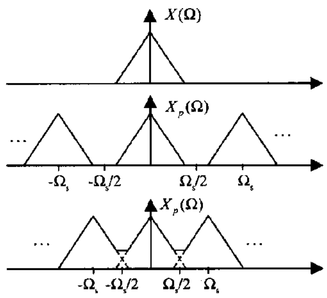
$$X_N[k] = \sum_{n=0}^{N-1} x_N[n] e^{-j2\pi nk/N} \qquad 0 \le k \le N$$

$$e^{j\phi} = \cos\phi + j\sin\phi$$

  - Direct computation of DFT: N^2 operations
- FFT: A fast algorithm to compute the DFT
  - $N\log_2 N$ operations
  - For 256 points window, FFT is ≈ 256/8 = 32 times faster than direct computation

5.5.2 The Sampling Theorem

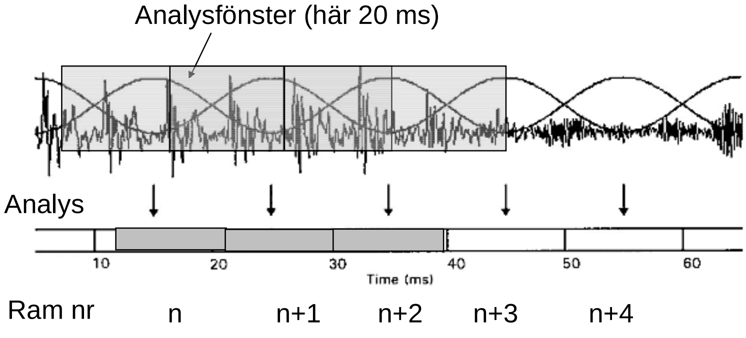
- The analog signal cannot be uniquely recovered from the digital signal if the analog signal has energy above the *Nyquist* frequency $F_s/2$
- Aliasing (Sw. vikningsdistorsion) will occur.
- An analog anti-aliasing low-pass filter is necessary



Ch 6 Speech Signal Representations

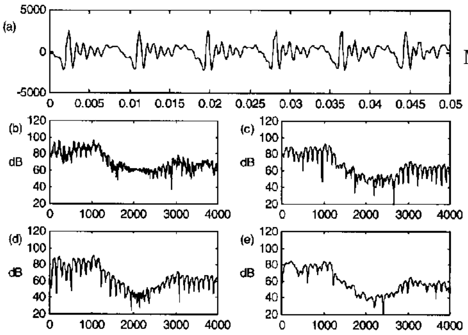
- Short-Time Fourier Analysis
  - Effect of different window functions
- Linear Predictive Coding
  - Spectral Analysis via LPC
  - Equivalent Representations
    - Reflection Coefficients
    - Log-Area Ratios
- Cepstral Processing
- Perceptually Motivated Representations
  - Mel-Frequency Cepstrum
  - Perceptual Linear Prediction (PLP)

Analys av signalen till en följd av korttidsspektra (ramar)



6.1 Short-Time Fourier Analysis

Effect of different window functions



Male voice /ah/   F0 = 110 Hz

Rectangular
30 ms

Rectangular
15 ms

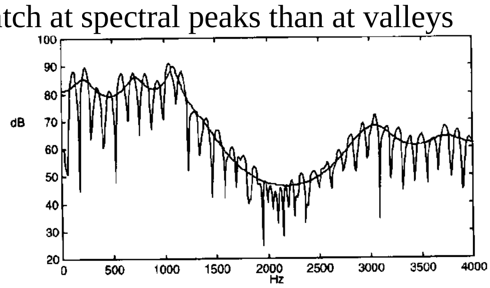
Hamming
30 ms

Hamming
15 ms

Window should be long enough to cover 2 pitch pulses
Short enough to capture short events and transitions

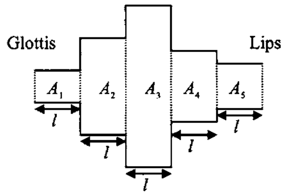
Linear Predictive Coding (LPC)

- Predicts the next sample as a linear combination of the past p samples $\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k]$
- Results in an all-pole filter which matches the signal spectrum
- Better match at spectral peaks than at valleys



LPC-based representations used for recognition

- Line Spectral Frequencies
  - popular in speech coding (GSM telephony)
- Reflection Coefficients
  - same as partial correlation coefficients (PARCOR)
- Log-Area Ratios
  - log of the ratio of the areas of adjacent sections of a lossless tube equivalent of the vocal tract
- LPC-Cepstrum



Perceptual Linear Prediction

- Transformation to the Bark frequency scale before computing the LPC coefficients
- Cubic root of energy instead of logarithm

RASTA

- Hermansky & Morgan “RASTA Processing of Speech”, IEEE Trans. On Speech and Audio Proc., 1994, **2**(4)
- Filtering (BP 2-10 Hz) of each channel amplitude in the short time spectrum
- Removes the filtering effect of the transmission channel
- Perceptually motivated

Cepstrumanalys

- Invers Fouriertransform av logaritmerat frekvensspektrum
- Bogert, Healy & Tukey (1963)*
- Ordlek: Spectrum-cepstrum, filtering-liftering,frequency-quefrency, phase-saphe
- Hög fonemdiskrimination (har det visat sig)
- Ortogonala koefficienter
- Grovstrukturen i spektrum beskrivs med ett litet antal parametrar
- Bra för grundtonsföljning

* “The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-autocovariance, Cross-Cepstrum and Saphe Cracking” Proc. Symp. Time Series Analysis, J. Wiley & Sons, 1963

Cepstral Processing

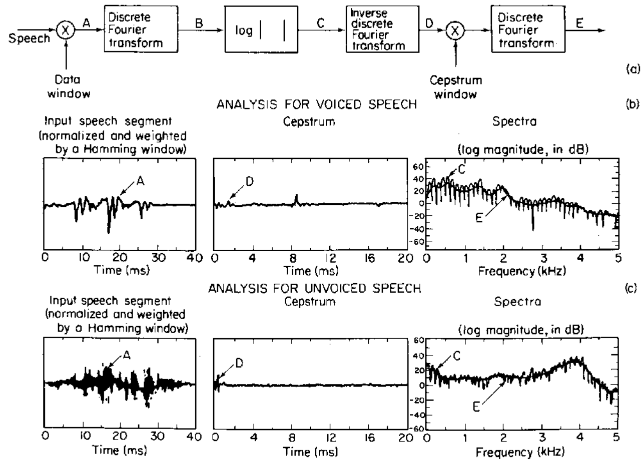
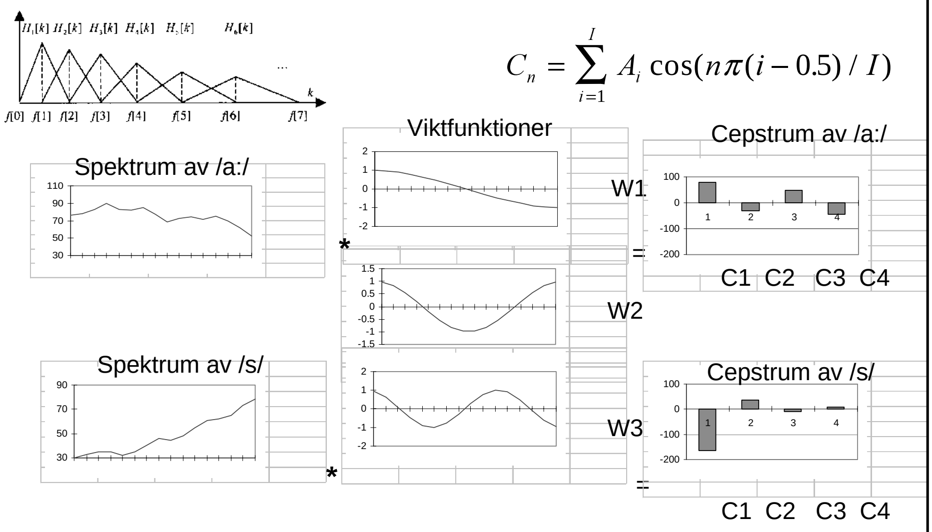


Fig. 10.20 (a) System for homomorphic analysis of speech; (b) analysis for voiced speech; (c) analysis for unvoiced speech.

From Oppenheim & Schafer, 1975

Mel-Frequency Cepstrum Coefficients (MFCC)

$$C_n = \sum_{i=1}^{I} A_i \cos(n\pi(i-0.5)/I)$$



Ch 7 Speech Coding

- Not included

September 19, 2003 Speech recognition course 2003 73

Ch 8 Hidden Markov Models

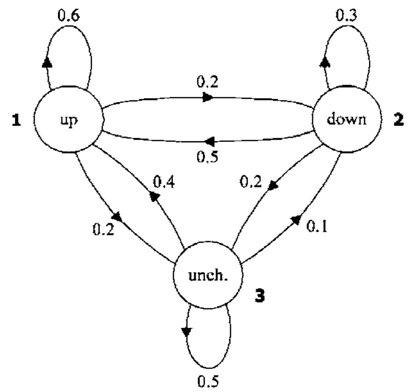
- The Markov chain
- Definition of the Hidden Markov Model
- Continuous and Semicontinuous HMMs
- Practical Issues in Using HMMs
- HMM Limitations

September 19, 2003 Speech recognition course 2003 74

8.1 The Markov Chain

- A Markov process
  - Can be in one of several states with random transition in state occupance
- **X**=$X_1, X_2, \ldots X_n$ is a sequence of random variables (states)
- First order Markov chain: $P(X_i \mid X_1^{i-1}) = P(X_i \mid X_{i-1})$
- The Markov assumption: The probability that the Markov chain will be in a particular state at a given time depends only on the state of the Markov chain at the previous time
- Parameters:
  - $a_{ij} = P(s_i=j|s_{i-1}=i)$ transition probability from state i to j
  - $\pi_i = P(s=i)$ initial probability

Fig 8.1 A Markov chain for the Dow Jones Industrial average



Initial state probability matrix

$$\boldsymbol{\pi} = (\pi_i) = \begin{pmatrix} 0.5 \\ 0.2 \\ 0.3 \end{pmatrix}$$

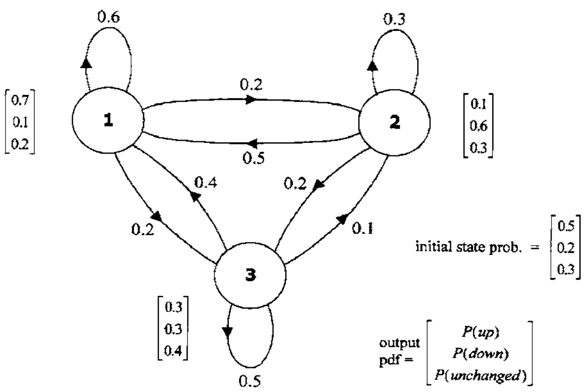
State-transition probability matrix

$$\mathbf{A} = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

8.2 Def. of the Hidden Markov Model

- A Markov chain where the state cannot be observed
- The observation is a probabilistic function of the state
- Defined by
  - $\mathbf{O} = \{o_1, o_2, \ldots o_M\}$ - An observation alphabet
  - $\Omega = \{1,2,\ldots,N\}$ - A set of states
  - $\mathbf{A} = \{a_{ij}\}$ - A transition probability matrix
    a_{ij} is the probability of transition state i to j
  - $\mathbf{B} = \{b_i(k)\}$ - An output probability matrix
    $b_i(k)$: probability of emitting o_k in state i
  - $\pi = \{\pi_i\}$ - An initial probability distribution
  - $\mathbf{\Phi = (A, B, \pi)}$ **-** The parameter set of an HMM
- Output independence assumption: the probability of emitting a symbol depends only on the state, not on previous observations

Fig 8.2 A hidden Markov model for the Dow Jones industrial average



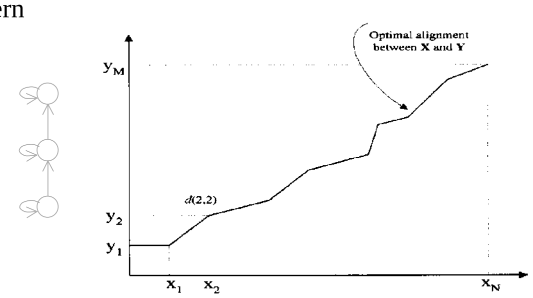
The states have no deterministic meaning

HMM - Three basic problems

- The Evaluation Problem
  - What is the probability that the model generates the observations?
- The Decoding Problem
  - What is the most likely state sequence in the model that produces the observations?
- The Learning Problem
  - How to adjust the model parameters to maximize the probability of producing the training data?

8.2.1 Dynamic Programming and DTW

- DTW and HMM are closely related
  - DTW computes spectral distance between two template patterns
  - In HMM a likelihood is computed that the model has produced the observed pattern



  - Many similarities (particularly DTW - Viterbi decoding)

8.2.2 How to Evaluate an HMM - The Forward Algorithm

- Calculate the probability (likelihood) of the observation sequence given the model
- Compute the probability for every possible state sequence
- Add them up

$$P(\mathbf{X}|\mathbf{\Phi}) = \sum_{\text{all}\mathbf{S}} P(\mathbf{S}|\mathbf{\Phi})P(\mathbf{X}|\mathbf{S},\mathbf{\Phi})$$

$$P(\mathbf{S}|\mathbf{\Phi}) = P(s_1|\mathbf{\Phi})\prod_{t=2}^{T} P(s_t|s_{t-1},\mathbf{\Phi}) = \pi_{s_1} a_{s_1 s_2} \cdots a_{s_{T-1} s_T}$$

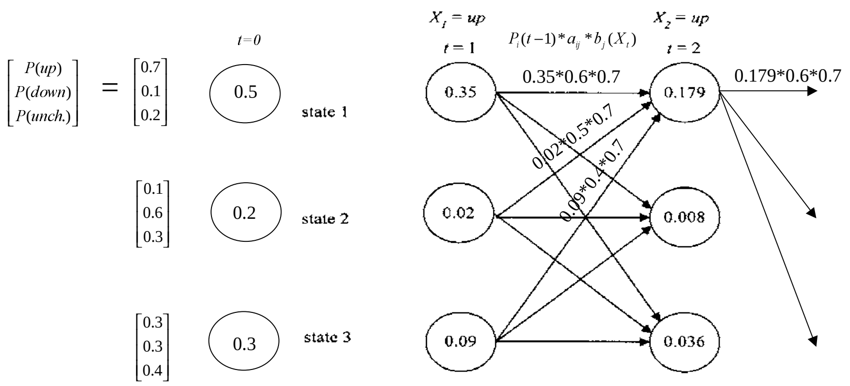
$$P(\mathbf{X}|\mathbf{S},\mathbf{\Phi}) = P(X_1^T|S_1^T,\mathbf{\Phi}) = \prod_{t=1}^{T} P(X_t|s_t,\mathbf{\Phi}) = b_{s_1}(X_1) b_{s_2}(X_2) \ldots b_{s_T}(X_T)$$

$$P(\mathbf{X}|\mathbf{\Phi}) = \sum_{\text{all}\mathbf{S}} a_{s_0 s_1} b_{s_1}(X_1) a_{s_1 s_2} b_{s_2}(X_2) \ldots a_{s_{T-1} s_T} b_{s_T}(X_T)$$

- Direct evaluation - very heavy computationally: $O(N^T)$ sequences
- Forward algorithm: fast since storing intermediate results: $O(N^2T)$
- Similar to DP

Forward probability $\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(X_t)$

Fig 8.4 Forward Trellis Computation



The forward trellis computation for the Dow Jones Industrial average

$$\mathbf{A} = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

Algorithm 8.2 The Forward Algorithm

ALGORITHM 8.2: ***THE FORWARD ALGORITHM***

Step 1: Initialization

$\alpha_1(i) = \pi_i b_i(\mathbf{X}_1) \qquad 1 \le i \le N$

Step 2: Induction

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(X_t) \qquad 2 \le t \le T;\ \ 1 \le j \le N \qquad (8.24)$$

Step 3: Termination

$P(\mathbf{X}|\Phi) = \sum_{i=1}^{N} \alpha_T(i)$ If it is required to end in the final state, $P(\mathbf{X}|\Phi) = \alpha_T(s_F)$

8.2.3 How to Decode an HMM - The Viterbi Algorithm

- Find the best path and calculate its probability
- Dynamic programming technique to lower the number of operations
- Very similar to the Forward algorithm (max instead of addition)

Viterbi-matchning mellan en HMM och ett yttrande

Förenklad algoritm för modellen nedan:

```
for(t=1; t<=T,++t)
   for(s=1; s<=S,++s)
      P(t,s) = P(O_t|S_s) * Max[P(t-1,s)*P_tr(s|s),  P(t-1,s-1)*P_tr(s|s-1),  P(t-1,s-2)*P_tr(s|s-2
return( P(T,S));
```

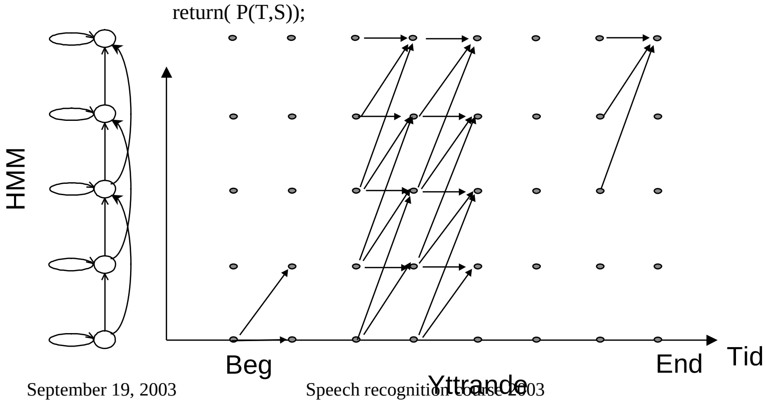
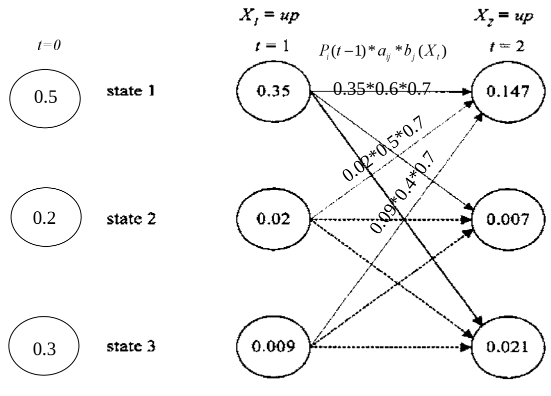


Fig 8.5 The Viterbi trellis computation

$$\begin{bmatrix} P(up) \\ P(down) \\ P(unch.) \end{bmatrix} = \begin{bmatrix} 0.7 \\ 0.1 \\ 0.2 \end{bmatrix}$$

$$\begin{bmatrix} 0.1 \\ 0.6 \\ 0.3 \end{bmatrix}$$

$$\begin{bmatrix} 0.3 \\ 0.3 \\ 0.4 \end{bmatrix}$$



$$\mathbf{A} = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.5 & 0.3 & 0.2 \\ 0.4 & 0.1 & 0.5 \end{bmatrix}$$

Alg. 8.3 The Viterbi Algorithm

ALGORITHM 8.3: *THE VITERBI ALGORITHM*

Step 1: Initialization

$V_1(i) = \pi_i b_i(X_1) \qquad 1 \le i \le N$

$B_1(i) = 0$

Step 2: Induction

$$V_t(j) = \underset{1 \le i \le N}{Max}\left[V_{t-1}(i) a_{ij}\right] b_j(X_t) \quad 2 \le t \le T; \quad 1 \le j \le N \qquad (8.25)$$

$$B_t(j) = \underset{1 \le i \le N}{Arg\max}\left[V_{t-1}(i) a_{ij}\right] \quad 2 \le t \le T; \quad 1 \le j \le N \qquad (8.26)$$

Step 3: Termination

The best score $= \underset{1 \le i \le N}{Max}\left[V_t(i)\right]$

$s_T^* = \underset{1 \le i \le N}{Arg\max}\left[B_T(i)\right]$

Step 4: Backtracking

$s_t^* = B_{t+1}(s_{t+1}^*) \quad t = T-1, T-2, \ldots, 1$

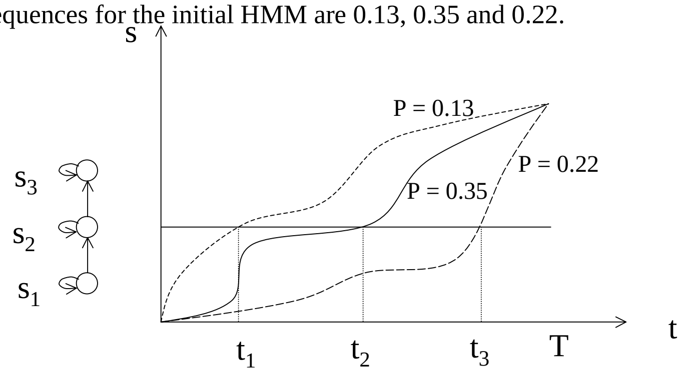
$\mathbf{S}^* = (s_1^*, s_2^*, \ldots, s_T^*)$ is the best sequence

8.2.4 How to Estimate HMM Parameters - Baum-Welch Algorithm

- The most difficult of the three HMM problems
- Solved by the iterative Baum-Welch algorithm (forward-backward)
- Unsupervised learning. Incomplete data. State sequence unknown.
  - Use the EM algorithm

Simplified illustration of EM estimation

Say, three paths have been found. The probabilities of the state sequences for the initial HMM are 0.13, 0.35 and 0.22.



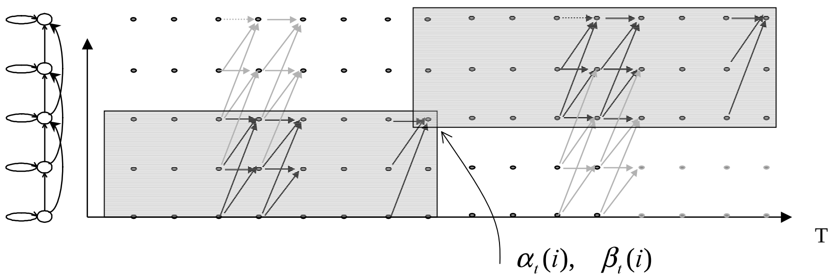
New $E(s_2) = (0.13\ X(t_1) + 0.35\ X\ (t_2) + 0.32\ X(t_3))\ /\ 0.70$

Not as simple as it may look, though

The Baum-Welch Algorithm - discrete HMM

- Forward probability $\alpha_t(i)$
  - The probability of generating a partial observation $X_1 \dots X_t$ ending at time *t* and state *i*
- Backward probability $\beta_t(i)$
  - The probability of generating a partial observation $X_{t+1} \dots X_T$ starting from time *t* and state *i. (Initialization?)*

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)\right] \qquad t = T-1, \dots, 1 \quad 1 \le i \le N$$



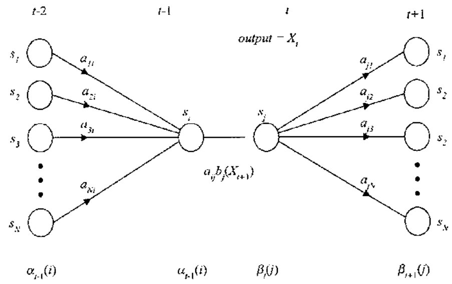
The Baum-Welch Algorithm (cont.)

Def $\gamma_t(i,j)$: The probability of the model having taken the transition from state i to state j at time t

$$\gamma_t(i,j) = P(\text{The model has switched from state } i \text{ to } j \text{ at time } t)$$

$$= \frac{\text{P(The model generates the observed sequence and switches from state i to j at time t)}}{\text{P(The model generates the observed sequence)}}$$

$$= \frac{\alpha_{t-1}(i)a_{ij}b_j(X_t)\beta_t(j)}{\sum_{k=1}^{N}\alpha_T(k)}$$



The Baum-Welch Algorithm (cont.)

New model estimates:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T}\gamma_t(i,j)}{\sum_{t=1}^{T}\sum_{k=1}^{N}\gamma_t(i,k)} \quad (8.40)$$

The ratio between the expected number of transitions from state i to j and the expected number of all transitions from state i

$$\hat{b}_j(k) = \frac{\sum_{t \in X_t = o_k}\sum_i \gamma_t(i,j)}{\sum_{t=1}^{T}\sum_i \gamma_t(i,j)} \quad (8.41)$$

The ratio between the expected number of times the observation data emitted from state j is o_k and the expected number of times any observation data is emitted from state j

Quite intuitive equations!